Visibility-Graph-based Shortest-Path Geographic Routing in Sensor Networks

Guang Tan, Marin Bertier, Anne-Marie Kermarrec

Visibility-Graph-based Shortest-Path Geographic Routing in Sensor Networks

Guang Tan Marin Bertier Anne-Marie Kermarrec
INRIA/IRISA, Rennes, France. Email: {guang.tan, marin.bertier, anne-marie.kermarrec}@irisa.fr

***Abstract*— We study the problem of shortest-path geographic routing in a static sensor network. Existing algorithms often make routing decisions based on node information in local neighborhoods. However, it is shown by Kuhn et al. that such a design constraint results in a highly undesirable lower bound for routing performance: if a best route has length c, then in the worst case a route produced by any localized algorithm has length $\Omega(c^2)$, which can be arbitrarily worse than the optimal. We present *VIGOR*, a VIsibility-Graph-based rOuting pRotocol that produces routes of length $\Theta(c)$. Our design is based on the construction of a much reduced *visibility graph*, which guides nodes to find near-optimal paths. The per-node protocol overheads in terms of state information and message transmission depend only on the complexity of the field's large topological features, rather than on the network size. Simulation results show that our protocol dramatically outperforms localized protocols such as GPSR and GOAFR+ in both average and worst cases, with reasonable extra overheads.**

I. Introduction

Geographic forwarding has been widely studied as a routing strategy for large wireless networks, mainly due to its simplicity and scalability. Typically, the routing algorithm greedily advance to nodes that are progressively closer to the destination. When reaching a local minimum, a face (or perimeter) routing method is employed to overcome the local minimum, and then the algorithm resumes greedy forwarding. For a network with uniform and dense sensor deployment in a regular region, such a method produces almost shortest paths with little overhead [12].

Such geographical algorithms, however, have serious efficiency problems in sensor fields with complex topologies. In many of the real-world environments the sensor field naturally has many obstacles/holes of arbitrary shape and size, causing the algorithm to run into local minima frequently. Yet the algorithms are often satisfied with only being able to escape those local minima, without regard to the efficiency of face routing. For instance, the GPSR protocol [12] always uses a right-hand rule to route around a hole, ignoring the possibility that there could be a much shorter face boundary path in the opposite direction. A series of subsequent work [13]–[15] has extended this strategy to offer shorter worst-case paths than GPSR. Unfortunately, as shown in [14], in the worst case a route produced by any localized algorithm has length $\Omega(c^2)$, c being the shortest possible path length. In a large-scale network where c can be at the order of tens or hundreds, the multiplicative factor c clearly means an excessive waste of energy and, very possibly, an unacceptable latency.

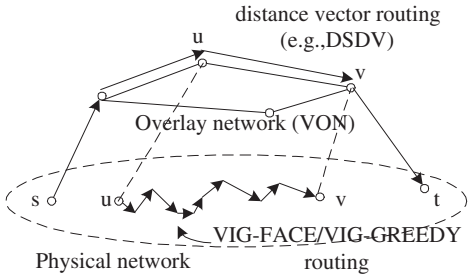


Fig. 1. Illustration for VIGOR. The overlay network VON runs a distance vector algorithm to find shortest Euclidean distance paths, while in the physical network, a face algorithm VIG-FACE (or its greedy variant VIG-GREEDY) realizes the routing between two overlay nodes. The face algorithm is designed in such a way that the hop count between two (visible) nodes u and v is within a constant factor of their Euclidean distance $|\overline{uv}|$ on the plane.

To improve this situation, a variety of optimization techniques have been proposed that use a limited amount of *non-local* information, for example by exploiting local face information [7], [16], or by using some kind of abstract representation of the field topology [4], [8], [10]. However, none of these heuristics provides any constant bound on path stretch. In this context, a natural question arises: Is it possible for the protocol to achieve $\Theta(c)$ performance while remaining scalable? Being scalable requires that the protocol's per-node overhead must be independent of, or grow very slowly with, the increase in network size N. This paper presents the first protocol that achieves this goal.

The protocol, called *VIGOR*, is based on *visibility graphs* (VGs), a structure in computational geometry that is often used to find shortest paths in an environment with obstacles. Using the information of detected hole boundaries, we develop techniques to construct a much reduced VG, which captures the field's topology in a form of size proportional to the complexity of the field's large geometric features (e.g., the number of holes above a certain size). The fact that such features are usually very small compared with N allows us to build a small overlay network, called a *Visibility-based Overlay Network* (VON), on top of the physical network. In the VON, nodes run a distance-vector based protocol (e.g., DSDV [18]) to route packets along paths of shortest Euclidean distance.

Within this framework, one of our major contributions is a face routing algorithm, called *VIG-FACE*, that routes between two visible VON nodes. (See Figure 1 for an illustration.) This algorithm is proved to produce paths with a constant hop

stretch between two nodes that are *visible* to each other. In conjunction with the distance vector algorithm at the overlay level, it enables VIGOR to achieve $\Theta(c)$ routing performance between general communication pairs. The per-node overhead depends only on the complexity of the large geometric features, rather than on the network size.

To improve the average-case efficiency of the protocol, we also develop a greedy version of VIG-FACE, which leads to the protocol VIGOR-G. Simulation results show that VIGOR-G produces near-optimal-length routes and dramatically outperforms protocols such as GPSR and GOAFR+ in both average and worst cases (by up to an order of magnitude), at low extra message and memory overheads.

II. Model and Preliminaries

A. Model

We assume that a total of N nodes, each with a distinct ID, are placed in the Euclidean plane $\mathbf{R}^2$, forming a connected network. Every node knows its own location, and a source knows the location of a destination through some location management system [6]. The network is planarized by some algorithm (e.g., CLDP [11]), so that no two edges cross each other in the graph. There is a maximum radio range, normalized to be 1, for all the nodes. The radio model does not need to be regular. $|\overline{uv}|$ denotes the Euclidean length of the line segment $\overline{uv}$; $D^G(u, v)$ denotes the shortest Euclidean path length between u and v on the graph G.

B. Visibility Graphs and Shortest Paths

It is shown [1] that the path of shortest Euclidean length between two points s and t in the plane avoiding polygonal holes/obstacles is a connected series of line segments, whose inner vertices are vertices of the holes. Consequently, any edge of a shortest path is a straight line segment e between two vertices p and q of holes. Since e does not cross any hole, p and q are "visible" from each other. The graph formed by all the hole vertices and these "visibility segments" is called the *Visibility Graph* (VG) of the polygonal environment; see Figure 2(a) for an example. In such a graph, the set of vertices that can be seen by a certain vertex p is called p's *visibility set*, denoted Φ_p. Likewise, a vertex p's *edge visibility set* is defined as the set of hole edges that can be seen by p, where an edge can be seen iff at least one point on that edge is visible. As such, the problem of finding shortest obstacle-avoiding path between two points reduces to how to construct the VG and how to search for the shortest path in the VG.

A *Reduced Visibility Graph* (RVG) technique can be used to reduce the number of edges in a VG. The line segment $\overline{xy}$ is a *tangent segment* iff its prolongation is tangent to the holes at both x and y. A RVG is a subset of VG that has all non-tangent segments removed (see Figure 2(b)). It can be proved that the RVG contains the shortest path between a pair of source and destination points as well [1].

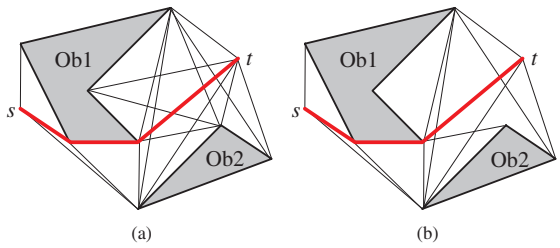


Fig. 2. An example of (a) visibility graph and (b) reduced visibility graph on a 2D plane. The gray regions are holes. The thick line segments (in red) represent the shortest path between s and t.

C. Visibility-Graph-based Routing

The basic idea of VG-based routing is to first identify the sensor nodes that represent the "hole vertices", and then organize those nodes into a *Visibility-based Overlay Network* (VON). If the VON is small enough in size, we can afford using a distance vector (e.g, the DSDV) protocol to maintain a shortest (Euclidean distance) overlay routing table at each VON node. When a source nodes wants to route to some destination node, it only needs to join the VON and route on the VON. The VON path can serve as a reference for finding a shortest path in the underlying network. Given the framework, we need to address two technical challenges: how to construct a small VON, and how to route between two VON nodes efficiently.

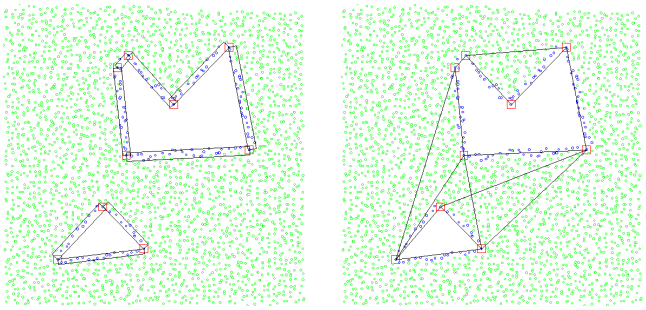
III. Constructing a Small VON

A. Detecting Holes

In a planar graph, a face can be viewed as a closed (polygonal) region represented by a sequence of nodes and directed edges. In this sequence, a node can appear more than once but edges cannot. The size of a face is defined as the number of edges in such a sequence. A face is said to be a *big face* if its size is greater than η, where η is a system parameter, and a *small face* otherwise. A big face is also called a *hole* for its great impact on routing performance. An example of holes is given in Figure 3(a), where $\eta = 15$ and the hole boundary nodes are shown in blue (darker color).

Each node p has a set of adjacent faces. For each face F_p in this set, p periodically routes a probing packet along F_p's boundary. This is done by routing to some fake point p' within p's interior angle at F_p and where no sensor node exists. Following the simple right-hand rule as widely used by geographic protocols such as GPSR and GOAFR+, such a packet will finally create a loop [12], which can be detected by p when the packet first returns. All the face probing packets are piggybacked with existing *keep-alive* beacons so no extra messages will be occurred.

To reduce message length, the face probing packets are dropped at a node that has a larger ID than the packet source node. As a result, only the node with the highest ID on a face F_p can receive its own packet. Such a node becomes F_p's *header*. The face probing packet can easily carry the number of hops it has completed, so that the face header can learn



(a) holes and bounding bars (b) the final VON

Fig. 3. The construction of a VON for a network. With $\eta = 15$, the VON has 9 nodes, in contrast with the original network size of 2658. The darker circles and squares represent hole boundary nodes and VON nodes, respectively. For clarity the out boundary nodes and their associated VON edges are not shown.

the face's size. If F_p's header finds that F_p is a big face, then it sends a second packet around F notifying all the nodes en route of F_p's size. After this, F_p's header can start to generate a *VON polygon* for F_p.

B. Constructing the VON Polygons

A hole's header node initially takes itself as the first VON node on its current hole, and then sends out a packet which will traverse the hole boundary in a clockwise direction. The packet has three fields: field 1 containing a sequence of locations of the VON nodes determined so far, field 2 containing a sequence of locations of all the nodes starting from last VON node up to current node, and field 3 containing a parameter $\omega > 0$. Every time a node becomes a new VON node, it first appends its location to field 1 of the packet, and then replaces the content of field 2 with its location, and finally forwards the packet to the next node. Upon receiving the packet, a node first appends its location to field 2 of the packet, and then checks whether the following two conditions hold: (1) there exists a rectangle that covers all the locations contained in field 2. The rectangle, called a *bounding bar*, has a maximum width ω, and an unconstrained length (see Figure 3(a) for an illustration); (2) the sequence of locations in field 2 has a monotonic sequence of projection points on the length-unconstrained sides of the bounding bar. If these two conditions are met, then the packet proceeds to the next node on the boundary; otherwise the current node becomes a VON node. When the packet returns to the hole header node, the header node can construct a sequence of edges from the location sequence contained in field 1 of the packet, which forms a VON polygon.

It is possible that the VON polygon generated has intersecting edges. In this case we need to make a finer approximation of the original hole by creating more VON nodes. Suppose that the header node has found two intersecting edges $\overline{v_1v_2}$ and $\overline{v_3v_4}$. (The case of more intersecting edges can be handled similarly.) It performs the above hole approximation procedure a second time, with two changes: (1) the locations of v_1, v_2, v_3 and v_4 are recorded in the packet as a new field, and (2) $\omega \leftarrow \omega/2$ in field 3. The newly added field tells that new VON nodes need be created only between v_1 and v_2, and between v_3 and v_4, so nodes outside the two ranges only need to forward the packet. The decreased ω makes a bounding bar less able to accommodate many nodes, so that a new VON edge spans fewer nodes, and new VON nodes can be created. Note that as ω tends to zero, the created VON polygon tends to overlap with the original hole boundary.

The intersection avoiding procedure is repeated until there is no intersection between the VON edges in the ranges (v_1, v_2) and (v_3, v_4). Finally, an intersection-free polygon will be created at the hole header node. This intersection-free property can be guaranteed since in the extreme case where the VON polygon reduces to the original hole boundary, the VON polygon will be free of intersection, because of the planar property of the underlying network graph. Our experiments show that self-intersection happens rarely when the initial ω is appropriately chosen (e.g., $\omega = 0.5$).

After determining a locally intersection-free VON polygon, a hole header node can broadcast the polygon (i.e., the sequence of its vertices) to the network. Note that the hole header node only needs to remember the VON nodes, rather than all the nodes, on its face. The broadcast at this stage, however, is not yet for constructing the final VON; instead it is only for detecting possible intersections between VON polygons created from nearby holes. Every hole header node periodically broadcasts its VON polygon information, and at the same time collects the broadcast messages from the network and checks if there are inter-hole intersections. Every time an intersection has been found, the hole header node performs the intersection avoiding procedure described earlier. The broadcast stops when a hole header node can find no intersection on its own VON polygon.

After all the above procedures, the hole header nodes can start broadcasting the VON polygons to the network – this time for completing the construction of the VON. Every node (not only the VON nodes) in the network maintains an edge set containing all the edges it can see, and updates the set each time it receives a broadcast message. From this edge set, a node can determine the set of VON nodes it can see, that is, its visibility set Φ. The visibility relationship between VON nodes directly translates to VON edges, which, along with the VON polygons already identified, constitute the VON. An example of VON is given in Figure 3(b).

C. Reducing the Number of VON Edges

In order to lower the message overhead of the distance vector protocol, we use Yao graph [21] to reduce the number of VON edges from $O(N_{von}^2)$ to $O(N_{von})$, where N_{von} denotes the number of nodes in the VON. The Yao graph with an integer parameter $k > 6$ is defined as follows. At each node v, draw k equally-separated rays from v which define k cones. In each cone, remove all edges, if there is any, except the shortest one incident on v; ties being broken arbitrarily. The resulting graph is known to be a spanner subgraph of the original graph [21].

Using the idea of Yao graph, we propose an algorithm called *VON-SPARSIFY* to reduce the number of VON edges. In this algorithm, when a node v picks a neighbor in a certain cone

C_v, it first checks whether C_v contains a visible node u such that $\overline{vu}$ is a tangent edge. Only when this condition holds does v establish a link to the nearest visible node u in C_v (here $\overline{vu}$ need not be tangent). Since non-tangent edges never appear in a shortest path in the VG, their removal will not affect the spanner property of the generated subgraph.

The Yao graph technique is also applied to non-VON nodes. For each non-VON node s, this will select at most k nodes out of s's visibility set Φ_s for establishing (virtual) links. These nodes define s's *selective visibility set*, denoted Φ'_s.

D. Reducing the Number of VON Polygons

Increasing the parameter η can reduce the number of VON polygons generated, yielding a smaller VON. For example, in Figure 3, increasing $\eta = 15$ to 40 will result in a smaller VON without the smaller triangular hole. No doubt there is a tradeoff between the size of VON and routing performance, which will be analyzed in the next section.

IV. The VIGOR Routing protocol

The VIGOR protocol integrates two components: the distance vector algorithm which is responsible of shortest path (in terms of Euclidean distance) routing on the VON, and a face algorithm that routes between two VON nodes in the physical network. Given a source node s and a destination node t, VIGOR first needs a procedure to determine whether the route needs to go through the VON, and if it does, at which node s' the route enters the VON and at which node t' it exits the VON.

A. Establishment of VON Routes

The node s first checks whether t is visible. It is easy to verify that t is visible to s iff $\overline{st}$ does not intersect any edge from s's edge visibility set. If t is found to be visible, then $s' = s$ and $t' = t$, which means that the route does not need to go through the VON; otherwise s needs some extra communication to determine s' and t'.

First, s sends its visibility set Φ_s (piggyback with its first packet) to t. Since a shortest path between s and t (or st-path for short) has not yet been established, the packet has to follow a sub-optimal path. This path can be generated using a simple existing algorithm such as GPSR or GOAFR+. We adopt the latter in the paper.

When t receives the packet containing Φ_s, it forwards the set to each member of t's selective visibility set Φ'_t. (Note that $|\Phi'_t| \leq k$.) Next, each $v \in \Phi'_t$ uses its VON routing table to select a VON node $u \in \Phi_s$ such that $D_v = |\overline{su}| + D^{von}(u, v) + |\overline{vt}|$ is the smallest. Here $D^{von}(u, v)$ means the shortest distance between u and v in the VON. Each v then sends a packet containing the chosen u and calculated D_v back to t. All these communications use the GOAFR+ algorithm.

Now, t picks the v such that D_v is the smallest, lets $t' = v$ and s' be the node u chosen by v, records t' and s' to a packet, and sends it to s – this time along the VON path. After receiving this packet, s knows s' and t'. In future transmissions, every packet from s will contain the nodes s' and t'. This information will be used by the distance vector algorithm for routing on the VON.

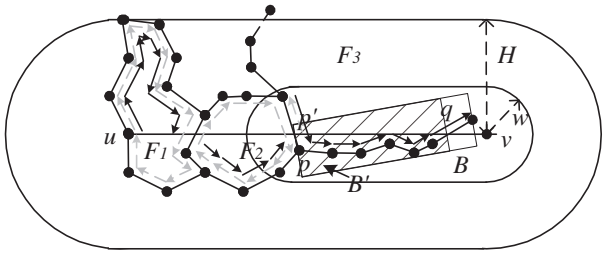


Fig. 4. The H-distance set that covers the nodes visited by VIG-FACE's routing between two visible VON nodes u and v. F_1 and F_2 are small faces and F_3 is a big face, B is a bounding bar of F_3, and B' is a rectangular sub-area of B that contains a path between p' and q. The small oval-shaped area is $\Psi(p', q, \omega)$.

B. Routing Between Two Visible Nodes

VIGOR uses a pure face routing algorithm to route from two visible nodes u and v. Here u and v can be the communication source/destination nodes or VON nodes. The algorithm uses a concept called the *H-Distance Set*. An H-Distance Set of a line segment $\overline{uv}$ on the plane, denoted $\Psi(u, v, H)$, is the set of points whose distance from $\overline{uv}$ is no greater than H. Here the distance of a point from $\overline{uv}$ is the smallest distance between that point and all points on $\overline{uv}$. Geometrically, $\Psi(u, v, H)$ corresponds to an oval-shaped area as illustrated in Figure 4.

The face routing algorithm of VIGOR, referred to as *VIG-FACE*, consists of the following steps.

Algorithm VIG-FACE (u, v)

Let p be the currently visited node; $p_{nearest}$ be the closest point to v the route has met so far on $\overline{uv}$; F_p be p's current face (i.e., p's adjacent face intersected by $\overline{p_{nearest}v}$); p_{next} be the next node to visit on F_p. While v is not directly reachable, repeat the following face routing process (in the clockwise direction by default):

1) If F_p is a big face, then start exploring the boundary of F_p. At each node, check whether the route will
   a) encounter a VON node other than v and p, or
   b) go beyond the boundary of $\Psi(u, v, \omega)$, or
   c) cross $\overline{uv}$ at a point that is farther from v than $p_{nearest}$.

   If so, turn back and explore the boundary of F_p in the opposite direction. Continue until reaching a node q such that $\overline{qq_{next}}$ intersects $\overline{uv}$ at a point p' closer to v than $p_{nearest}$. Let $p \leftarrow q$ if q_{next} does not lie on uv, or $p \leftarrow q_{next}$ otherwise; let $p_{nearest} \leftarrow p'$.
2) If F_p is not a big face, then explore the whole boundary of F_p, at the same time updating $p_{nearest}$; after returning to p, advance to the node q such that $\overline{qq_{next}}$ intersects line uv at $p_{nearest}$. Let $p \leftarrow q$ if q_{next} does not lie on uv, or $p \leftarrow q_{next}$ otherwise.

Figure 4 gives an example of the algorithm routing through three consecutive faces.

C. VIGOR is Correct

We examine VIG-FACE(u,v)'s behavior on the two types of faces. We first assume that the VON is correctly constructed; the handling of exceptions will be discussed later. When the current face F_p is not a big face, the algorithm ensures that a

positive progress will be made toward v after routing on F_p. When F_p is a big face, the following lemma shows that the algorithm will also make a progress toward v. (The proofs of all lemmas and theorems can be found in [19]).

Lemma 1: Suppose p' and q are two neighboring intersection points between the VON edge $\overline{uv}$ and a big face F_p^b. Then there exists at least one $p'q$-path on F_p^b's boundary that does not contain an intermediate VON node.

Suppose VIG-FACE(u,v) starts traversing the big face F_p^b from a node adjacent to p', that is, from a node p such that $\overline{pp_{next}}$ crosses $\overline{uv}$ at p' (or from p' itself if p' is a real node). Lemma 1 suggests that there must be a pq-path that does not contain an intermediate VON node, thus justifying the turning-back operation of VIG-FACE when it encounters a VON node other than v and p. Lemma 1 also implies that both p' and q are covered by the same bounding bar B. Because the boundary nodes in B have monotonic projections on B's length-unconstrained edges, this further means that there exists a $p'q$-path whose nodes are all covered by a shorter bar B', which is intersected by $\overline{uv}$ at p' and q, shown as a shaded area in Figure 4. It is possible to verify that $B' \subset \Psi(p', q, \omega) \subset \Psi(u, v, \omega)$. Hence, there exists a $p'q$-path on F_p^b's boundary whose nodes are all covered by $\Psi(u, v, \omega)$. This explains why VIG-FACE turns back when it hits the boundary of $\Psi(u, v, \omega)$. Now that a path leading to q is guaranteed to be found in the region scoped by $\Psi(u, v, \omega)$, VIG-FACE can finally make its way to q, making a positive progress toward v.

In summary, VIG-FACE makes a positive progress toward v every time it completes executing on a face. This ensures that v will be finally reached because there are a finite number of faces in the network. Together with the correctness of the distance vector algorithm (e.g., [18]) this establishes the correctness of VIGOR.

VIGOR's correctness is robust to network dynamics. In normal cases, for VIG-FACE(u, v) the two nodes u and v are visible. When nodes fail or new nodes are added to the network, v may be no longer visible to u, or may lose its role as a VON node. This will by no means affect the correctness of VIG-FACE, since the algorithm will still follow the principle of face routing – traverse a face, make a progress, traverse the next face, and so on. The only thing affected is the direction chosen for face routing, which may mislead the packet into longer (but viable) face boundary paths and result in suboptimal performance.

In the case that the node v fails or becomes unreachable, the algorithm can detect this by checking whether it has completed traversing a face without getting closer to v. The routing error will be reported to u, which can find an alternative VON route in its routing table and try routing again.

D. VIGOR Has Constant Stretch

In this section we analyze the performance of VIGOR. We adopt the widely used unit disk graph (UDG) model [12] to represent the network. (This assumption is needed only in this section.) To avoid pathologic network topologies (e.g., one in which a node has $O(N)$ degree), we employ an alternative planarization algorithm developed by Wang and Li [20] that guarantees a bounded degree χ for every node. Such a localized algorithm also generates a *spanner*, meaning that for any two nodes, their shortest path length in the subgraph is within a constant factor δ, called the *stretch factor*, of their shortest path length in the original graph. The resulting graph is referred to as a *WL graph*. The main result is the following theorem (see a complete proof in [19]).

Theorem 1: Let $L^{VIG}(s,t)$ be the hop count needed by VIGOR to route from node s to node t, and $L^{OPT}(s,t)$ be the hop count of a shortest possible path between s and t in the network, then

$$L^{VIG}(s,t) \leq \frac{64\delta(\chi+1)(H+1)(H+1+2/\pi)}{1-2\sin(\pi/k)} L^{OPT}(s,t),$$

where $H = \max\{\omega, \eta/2\}$.

Proof: (Sketch) We first consider the case where both s and t are not within any VON polygons, which suffices to show the main idea of the proof. Consider the non-trivial case in which $|\overline{st}| > 1$.

First, we look at the algorithm's behavior between two adjacent VON nodes (or between an endpoint and a VON node). Again consider the non-trivial case $|\overline{uv}| > 1$. For a big face F^b, we already know from the correctness analysis that the algorithm can route around it without going beyond the region scoped by $\Psi(u, v, \omega)$. For a small face F^s, which has at most η boundary edges (and hence a perimeter at most η), the algorithm can go at most $\eta/2$ far away from $\overline{uv}$, because otherwise F^s would have a perimeter greater than η. Thus, $\Psi(u, v, \eta/2)$ suffices to bound the region visited by the algorithm when exploring a small face. Combining these two cases gives that the nodes visited by the algorithm while it routes between u and v are bounded by the H-distance set of $\overline{uv}$, where $H = \max\{\omega, \eta/2\}$. Now we use a technique due to Kuhn et al. [13] (Lemma 4.1) to bound the number of nodes within that region. Applying this result to $\Psi(u, v, \max\{\omega, \eta/2\})$ yields an upper bound of the number of nodes visited by VIG-FACE:

$$N^{VIG}(u,v) \leq 16(\chi+1)(H+1)\Big(H+1+\frac{2}{\pi}\Big)|\overline{uv}|,$$

where $H = \max\{\omega, \eta/2\}$.

From the description of VIG-FACE, we can see that the algorithm visits any node of a small face at most twice. Also, the $p_{nearest}$ has the effect of preventing the algorithm from traversing any edge of a big face more than twice. Thus for any particular face, the algorithm visits any node at most two times. Considering the fact that a node can belong to at most two adjacent faces that are intersected by the line $\overline{uv}$, we can conclude that a node in the region $\Psi(u, v, \max\{\omega, \eta/2\})$ will be visited at most four times by the algorithm. Therefore, the hop count needed by VIG-FACE(u, v) is:

$$L^{VIG}(u,v) \quad \leq \quad 4N^{VIG}(u,v)$$

where $H = \max\{\omega, \eta/2\}$.

Next, consider an st-path that *avoids* the VON polygons; that is, does not contain a point lying inside any VON

polygons (but can touch the boundary of the VON polygons). We denote its minimum Euclidean length by $D^{\mathbf{R}^2}(s,t)$. Let G_{von} be the graph corresponding to the VON as defined in Section III, G_{von+} be G_{von} augmented by adding s and t, as well as the edges determined by the visibility relationship between s, t and all the VON nodes, to G_{von}. Applying the VON-SPARSIFY algorithm to G_{von} and G_{von+} generates two subgraphs, YG_{von} and YG_{von+}, respectively. By combining Clarkson's induction technique [5] with Lukovszki's result [17] (Lemma 2.2), we can prove that $D^{YG_{von+}}(s,t) \leq \frac{1}{1-2\sin(\pi/k)} D^{\mathbf{R}^2}(s,t)$.

When VIGOR determines s's VON entry and exit nodes , it actually finds a shortest st-path via the edges in the union of YG_{von+}'s edge set and the set of visibility edges between s and all nodes in Φ_s, which is a superset of YG_{von+}'s edge set. Since the addition of edges to a graph does not increase the shortest path length between any two nodes in that graph, the shortest st-path length found by VIGOR, denoted $D^{VIG}(s,t)$, is no greater than the shortest st-path in YG_{von+}. Hence, $D^{VIG}(s,t) \leq D^{YG_{von+}}(s,t) \leq \frac{1}{1-2\sin(\pi/k)} D^{\mathbf{R}^2}(s,t)$.

Let $\mathcal{P}(s,t)$ be the edge set of the VON path from s to t, then we have

$$\begin{aligned} & L^{VIG}(s,t) \\ &= \sum_{(u,v)\in\mathcal{P}(s,t)} L^{VIG}(u,v) \\ &\leq 64(\chi+1)(H+1)(H+1+2/\pi) \sum_{(u,v)\in\mathcal{P}(s,t)} |\overline{uv}| \\ &= 64(\chi+1)(H+1)(H+1+2/\pi) D^{VIG}(s,t) \\ &\leq \frac{64(\chi+1)(H+1)(H+1+2/\pi)}{1-2\sin(\pi/k)} D^{\mathbf{R}^2}(s,t), \end{aligned} \tag{1}$$

where $H = \max\{\omega, \eta/2\}$.

Last, consider a shortest st-path on the WL-graph, whose Euclidean length is denoted by $D^{WL}(s,t)$. We can prove that $D^{\mathbf{R}^2}(s,t) \leq D^{WL}(s,t)$. Denote the minimum length of an st-path on the UDG by $D^{UDG}(s,t)$. Then we have

$$D^{\mathbf{R}^2}(s,t) \leq D^{WL}(s,t) \leq \delta D^{UDG}(s,t) \leq \delta L^{OPT}(s,t)$$

where δ is the stretch factor of the WL-graph. Combined with (1), this proves the theorem.

If either s or t is within some VON polygon, the VON routing needs to be slightly modified in order to ensure optimality. The details and proof are omitted here. ■

The stretch result presents a salient feature of our protocol. To the best of our knowledge, no prior geographic routing algorithm has achieved a constant bound (even in a UDG graph). Note that the correctness of our protocol does not need this UDG assumption.

V. VIGOR-G: A GREEDY+FACE PROTOCOL

In this section, we present a protocol, named *VIGOR-G*, that combines greedy routing and face routing. This protocol also runs on a planar (not necessarily a UDG) network.

VIGOR-G works in the same way as VIGOR except that VIGOR-G incorporates a greedy algorithm, referred to as *VIG-GREEDY*, to route between two visible VON nodes. This algorithm tries to route greedily toward the destination v whenever possible, that is, by forwarding the message to the neighbor located closest to v. When reaching a local minimum with respect to v, the algorithm employs a face routing to overcome the local minimum.

When VIG-GREEDY reaches a local minimum at a node p, it uses the following two rules to determine the "right" face routing direction based on the information of p's *associated VON edge* $\overline{P^{ccw}P^{cw}}$, where P^{ccw} and P^{cw} represent p's VON node neighbors counter-clockwise and clockwise on the face, respectively. (If more than one such edges exist in the case of a complex VON polygon, then one is picked randomly.) Note that P^{ccw} and P^{cw} both belong to p's visibility set.

1) *Direction Rule 1*: If p is not a VON node, then calculate its projection point p' on the line $P^{ccw}P^{cw}$. If $\overline{pv}$ is angularly closer to $\overline{p'P^{ccw}}$ than to $\overline{p'P^{cw}}$, then take the counter-clockwise direction, otherwise take the clockwise direction. In Figure 5(a), the grey half disk represents the angular range for $\overline{pv}$ to choose a counter-clockwise direction.
2) *Direction Rule 2*: If p is a VON node, then if $\overline{pv}$ is angularly closer to $\overline{pP^{ccw}}$ than to $\overline{pP^{cw}}$ (Figure 5(b)), take the counter-clockwise direction; otherwise take the clockwise direction.

It is possible that the line pv crosses a VON polygon (Figure 5(c)), which may lead the packet into the wrong direction and create a big detour. VIG-GREEDY determines whether this is happening by checking whether $\overline{pv}$ intersects any of p's associated VON edges. For a particular VON edge $\overline{P^{ccw}P^{cw}}$, if p is on the right of this edge, it means that p is inside the VON polygon that contains $\overline{P^{ccw}P^{cw}}$; in this case the algorithm first calculates p's projection p' on $P^{ccw}P^{cw}$ and then checks whether $\overline{p'v}$ intersects $\overline{P^{ccw}P^{cw}}$. The above procedure is performed every time the algorithm hits a big face boundary node except during the face routing. The rule for choosing a direction in this case is as follows:

Direction Rule 3: If $\overline{pv}$ (or $\overline{p'v}$ in the case that p is within the VON polygon that contains $\overline{P^{ccw}P^{cw}}$) intersects $\overline{P^{ccw}P^{cw}}$, then take a counter-clockwise direction if p is on the left of the line uv, otherwise take a clockwise direction.

After determining the face routing direction, the algorithm routes along the face boundary until a node q such that $\overline{qq_{next}}$ intersect $\overline{pv}$ at some point closer to v than p.

When VIG-GREEDY starts face routing at node p, the next face to traverse is the face intersected by $\overline{pv}$, where v is next VON node on the path. This guarantees that a positive progress toward v is made after every face routing procedure. According to Frey and Stojmenovic's result (Theorem 4) [9], the face routing procedure along with the greedy steps' progressive forwarding can guarantee delivery in a planar graph. Together with the correctness of the distance vector algorithm, this means that VIGOR-G is correct in general planar graphs. VIGOR-G handles network dynamics or routing exceptions in

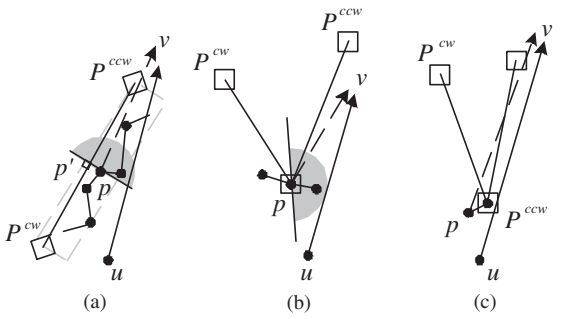


Fig. 5. Heuristics for determining face routing direction in VIG-GREEDY. The squares represent VON nodes. u and v are the source and destination nodes, p is the current node, p' is the projection node of p on the line $P^{ccw}P^{cw}$.

the same way as VIGOR. Executed with an incorrect VON, it behaves like a traditional greedy+face protocol with face routing directions chosen in a sub-optimal way.

VI. Simulations

In this section we evaluate the performance of VIGOR-G through simulations. We focus on the algorithm's behavior on the topological level. Comparisons are made with GPSR [12], GOAFR+ [13], and a centralized shortest-path algorithm, which serves as a baseline. The sensor network is deployed in a $1000m \times 1000m$ square field with irregular holes. Figure 6 shows four example field layouts. The number of sensors ranges from 2958-3732. Sensor nodes are placed on a grid and then perturbed with a random shift following a normal distribution [4]. The generated networks remain connected. For the construction of VONs, the Yao-graph parameter k is set to 7. The maximum width of a bounding bar ω is 30 meters. The threshold size for a face to be a hole, η, is set to 15 by default. Each sensor has a communication range of 60 meters. We have considered both UDG and quasi-UDG radio models. Since the advantage of VIGOR/VIGOR-G comes from its capturing large topological features, local variation in connectivity has little effect on its global behavior, and thus does not fundamentally change the comparative results. We therefore only report on the case of UDG radio model.

Hop Stretch We randomly pick 3000 pairs of source and destination nodes in the network and run the four routing algorithms for those pairs. For each pair, the path length in terms of hop count produced by the centralized algorithm is taken as the baseline, and the other three algorithms are compared in terms of path stretch, that is, the ratio of a path's length to the baseline value. Figure 7 plots the stretch value distributions of the algorithms under the four field layouts in Figure 6.

All the four plots show that VIGOR-G has a significant smaller stretch than GPSR and GOAFR+. For instance, in Figure 7(b), VIGOR-G has a mean stretch of 1.05, in contrast with GPSR's 4.10 and GOAFR+'s 5.26. This indicates that VIGOR-G achieves near-optimal performance in average case, and considerably outperforms the other two representative algorithms.

The worse-case behavior of the algorithms can be observed from the maximum stretch values. With this metric, an even bigger difference can be found between these algorithms. Throughout the experiments, the stretch of VIGOR-G remains below 2.0, while the other two have a stretch varying widely. For instance, in Figure 7(b), GPSR produces a stretch as high as 56.17, nearly 30 times that of VIGOR-G's peak value of 1.91. In other cases, GPSR and GOAFR+ consistently produce a maximum stretch nearly an order of magnitude higher than that of VIGOR-G. The great advantage of VIGOR-G in the worst case also suggests that our heuristics are successful.

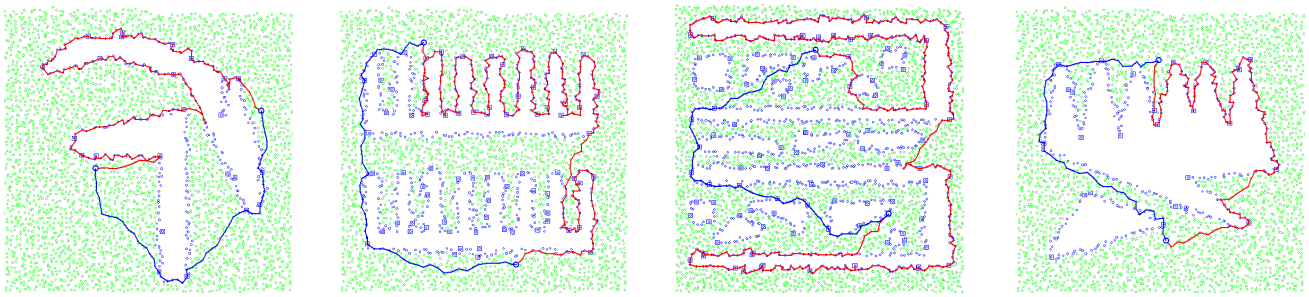
Protocol Message Overhead In addition to the keep-alive beacons between neighbors, the VIGOR-G protocol requires extra messages to (re)construct the VON, broadcast the VON polygons, and to run the distance vector algorithm on the VON. Assuming an inter-node beacon interval of $b = 1$ second [3], [12] and taking the beacon message overhead as a baseline, we simulate the maintenance of VON and compare the protocol's overhead with the baseline.

The maintenance of VON involves all the boundary nodes of holes. Since face probing packets are all piggybacked with the keep-alive beacons, we only need to consider the packet issued by hole header nodes. We assume that a hole header reconstructs its VON polygon and broadcast the polygon every b_1 time, where b_1 is adjustable. For the message overhead of the distance vector algorithm running on the VON, we take the DSDV protocol [18] as a running example and borrow the experimental setting from [3]. In [3], the periodic route update interval for DSDV is set to 15 seconds. Taking into account the triggered updates, the effective rate of protocol message transmission is one update per node per second. We let the update messages be transmitted over the VON edges, which correspond to the paths generated by VIGOR-G in the underlying network.

Table I shows the protocol overhead measured in number of messages per second for varying b_1; the baseline overhead is given in the parenthesis. It can be seen that for all the settings, VIGOR-G has an overhead within a moderate factor of the baseline. For instance, for the field topology Figure 6(a) with $b_1 = 15$ (the same as the VON route update interval), the protocol overhead is 4849 messages per second, which translates to 1.4 messages per node per second. We believe this is a reasonable overhead for a single node.

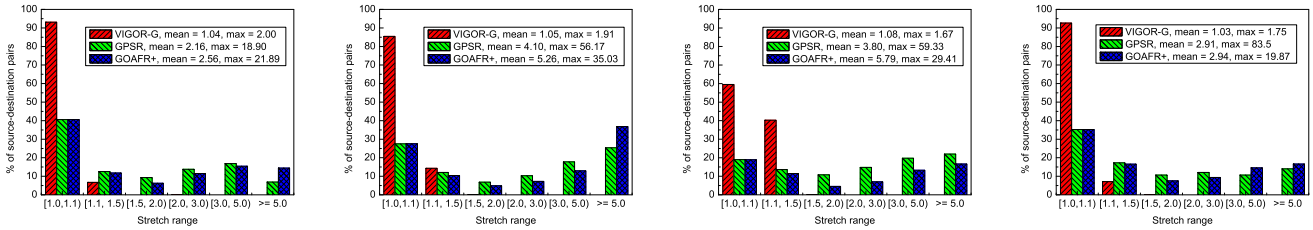
It should also be noted that the settings in [3] are for a dynamic network with high node mobility. In a static setting, the update of the VON structure and the VON routing tables can be performed at a much lower frequency, which leads to an even smaller protocol overhead.

Memory Requirement VIGOR-G requires every VON node to maintain a routing table, which is of size $O(N_{von})$. In our experiments, the sizes of VONs are at the order of tens. Every non-VON node in the network is also required to remember an edge visibility set (which also contains its node visibility information). Theoretically, this set is of size $O(N_{von})$; in practice, it is usually much smaller than N_{von}.



(a) $N = 3356$, avg degree = 10.1 (b) $N = 3166$, avg degree = 9.6 (c) $N = 3732$, avg degree = 9.32 (d) $N = 2958$, avg degree = 10.0

Fig. 6. Four $1000m \times 1000m$ sensor fields with holes. The red and blue lines show several examples of paths generated by GPSR and VIGOR-G, respectively.



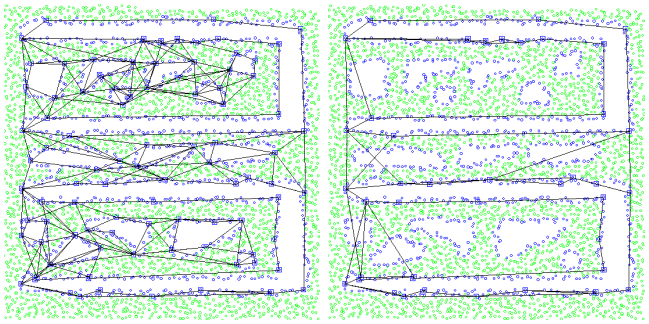
(a) Stretch distribution: Fig.6(a) (b) Stretch distribution: Fig.6(b) (c) Stretch distribution: Fig.6(c) (d) Stretch distribution: Fig.6(d)

Fig. 7. Path stretch distributions of VIGOR-G, GPSR, and GOAFR+ under the field layouts in Figure 6.

	$b_1 = 10$	$b_1 = 15$	$b_1 = 20$	$b_1 = 30$
Fig. 6(a)	5084(3356)	4849(3356)	4731(3356)	4613(3356)
Fig. 6(b)	5797(3166)	5561(3166)	5443(3166)	5325(3166)
Fig. 6(c)	8553(3732)	8264(3732)	8120(3732)	7976(3732)
Fig. 6(d)	4780(2958)	4569(2958)	4464(2958)	4358(2958)

TABLE I

NUMBER OF PROTOCOL MESSAGES PER SECOND AS COMPARED WITH THE BASELINE BEACON MESSAGE OVERHEAD (IN PARENTHESIS).



(a) VON with $\eta = 15$ (b) VON with $\eta = 55$

Fig. 8. Effect of η on the VON.

For instance, in our experiments, a non-VON node needs to remember at most 21 visible edges, each represented by two points, in the four field layouts in Figure 6. These requirements are unlikely to be a concern for current sensor hardware.

Effect of the Parameter η The parameter η determines whether a face is a hole, thus needs to participate in the VON. Intuitively, the smaller η is, the more VON polygons there will be in the VON, and the closer the generated path length will be to the optimal. On the other hand, the larger VON requires more messages transmission and memory to maintain. Figure 8 shows the VON constructed for the field layout Figure 6(c) under different η. When $\eta = 15$, the VON has a size of 92; increasing η to 55 yields a much sparser VON of only 49 nodes; see Figure 8(b).

We vary η to examine the tradeoff between path quality and protocol overhead in terms of N_{von} and the total number of VON maintenance messages. The trend agrees well with the intuition discussed above. For example, when $\eta = 25$, the average stretch is 1.083; for $\eta = 100$, the average stretch increases slightly to 1.130. For these two cases, the message overhead is 8162 and 7136 messages per second, respectively. The results suggest that allowing only a small extra overhead can significantly improve routing performance as compared with a localized algorithm.

VII. DISCUSSION

Scaling Issues The per-node protocol overhead of VIGOR(-G) is $O(N_{von})$, where N_{von} is the number of VON nodes. N_{von} indeed reflects the field's geometric complexity, embodied by the number, size, and shape of holes above a certain size. What justifies the VIGOR protocol is the fact that in real-world environments, the geometric complexity is often low enough compared with the network size, allowing the protocol to run without overloading sensor nodes. For example, the number of buildings, or the number of boundary edges of buildings, in a factory or a campus, is usually very small compared with the number of sensors that could be deployed in such environments. For even more geometrically complex

fields and larger systems, choosing suitable parameters η and ω provides a convenient way to tradeoff between protocol overheads and path quality.

Network Dynamics We expect that in a static sensor network, topology changes are usually local and happen infrequently (due to, e.g., depletion of energy or external damages), so salient, global features like large holes in the communication graph are rather stable under those local changes. This justifies a low-frequency update of the visibility graph which leads to low overheads. Even if the VON becomes very inaccurate due to massive network errors, only the performance will be affected; the protocol will remain correct.

VIII. Related Work

Many efforts have been made to improve the performance of localized face routing algorithms or their greedy variants. Kuhn et al. [14] propose a localized algorithm that can achieve asymptotically optimal performance in a UDG network. In [13], they further propose GOAFR+, a protocol that achieves good average-case performance while preserving worst-case optimality. In [14], Kuhn et al. prove that any localized algorithm can produce path length quadratic of the optimal, which is undesirable in a large-scale network. Thus, in order to guarantee finding a reasonably short path, using a certain amount of non-local information is inevitable. In this direction, Leong et al. [16] have proposed Path Vector Face Routing, which achieves superior performance to GPSR by having nodes remember some information about their local faces. A similar approach has also been suggested in [7] to support "path migration" and improve path quality. Both sets of research, however, only consider optimization on local faces. When it comes to a network with faces of arbitrarily complex shape and large size, the limited horizon makes them unable to find a globally short path.

In [2], Arad et al. present a Node Elevation Ad-hoc Routing (NEAR) scheme that assigns virtual coordinates to nodes so that local minima can be avoided more efficiently. Very recently, Jiang et al. [10] propose a novel information model in which a hole and its affected nodes are identified as an unsafe area. In so doing the algorithm can save many steps by avoiding entering an area that does not contain a path to the destination. While these heuristics are shown to be very beneficial for routing performance, neither is proved to provide a bound on path stretch.

Bruck et al. [4] propose MAP, a naming and geographic routing protocol that produces globally short paths using medial axis graphs (MAGs). After a preprocessing stage, a MAG is stored at each node in the network. With MAG, routing is first planned on the abstract medial axis graph, and then realized in each canonical region. In [8], a protocol called GLIDER with a similar principle to that of MAP is proposed. MAP and GLIDER share several features with VIGOR: they all try to capture the field's large geometric and topological features in a succinct form, and the routing is performed at two levels: the abstraction level, and the level of underlying networks. One major difference is that VIGOR focuses on shortest path routing and has provable performance, whereas MAP and GLIDER have other focuses and provide no guarantee in this regard.

IX. Conclusions

We have presented visibility-graph-based geographic protocols for shortest path routing in sensor networks. VGs facilitate routing through topologically complex environments along paths with shortest Euclidean lengths. The proposed protocols show excellent performance with low extra overheads in addition to the requirement by basic geographic routing protocols such as GPSR.

Acknowledgment

We would like to thank Davide Frey and Cigdem Sengul for comments on early drafts of the paper. We also thank the anonymous reviewers for their helpful feedback.

References

[1] H. Alt and E. Welzl. Visibility graphs and obstacle-avoiding shortest paths. *Zor-Zeitschrift fur Operation Research* 32:145-164, 1988.
[2] N. Arad and Y. Shavitt. Minimizing recovery state in geographic ad-hoc routing. *Proc. of MobiHoc'06*.
[3] J. Broch, D A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. *Proc. of MobiCom 1998*.
[4] J. Bruck, J. Gao, A. Jiang. MAP: Medial Axis Based Geometric Routing in Sensor Networks, *Proc. of MobiCom 2005*.
[5] K. L. Clarkson. Approximation Algorithms for Shortest Path Motion Planning. *Proc. of ACM STOC 1987*.
[6] S. M. Das, H, Pucha, and Y. C. Hu. Performance Comparison of Scalable Location Services for Geographic Ad Hoc Routing. In *INFOCOM 2005*.
[7] Q. Fang, J. Gao, L. J. Guibas. Locating and Bypassing Routing Holes in Sensor Networks, *Proc. of INFOCOM 2004*.
[8] Q. Fang, J. Gao, L. Guibas, V. de Silva and L. Zhang, GLIDER: Gradient landmark-based distributed routing for sensor networks, *INFOCOM 2005*.
[9] H. Frey and I. Stojmenovic. On Delivery Guarantees of Face and Combined Greedy-Face Routing in Ad Hoc and Sensor Networks. *Proc. of MobiCom 2006*.
[10] Z. Jiang, J. Ma, W. Lou, and J. Wu. An information model for geographic greedy forwarding in wireless ad-hoc sensor networks. *Proc. of INFOCOM'08*.
[11] Y-J Kim, R. Govindan, B. Karp, and S. Shenker. Geographic Routing Made Practical. *Proc. of NSDI 2005*.
[12] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. of MobiCom 2000*.
[13] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad-hoc routing: Of theory and practice. *Proc. of PODC'03*.
[14] F. Kuhn, R. Wattenhofer, and A. Zollinger. A Asymptotically Optimal Geometric Mobile Ad Hoc Routing. *DIALM-POMC'02*.
[15] F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *MobiHoc'03*.
[16] B. Leong, S. Mitra, and B. Liskov. Path vector face routing: Geographic routing with local face information. In *ICNP 2005*.
[17] T. Lukovszki. New Results on Geometric Spanners and Their Applications. Ph.D. thesis, Univ. of Paderborn, 1999.
[18] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *MobiCom'94*.
[19] G. Tan, M. Bertier, and A-M. Kermarrec. Visibility-Graph-based Shortest-Path Geographic Routing in Sensor Networks. Technical Report INRIA, France. http://www.irisa.fr/ asap/Members/gtan/visibility.pdf
[20] Y. Wang, X. Li. Localized Construction of Bounded Degree and Planar Spanner for Wireless Ad Hoc Networks. *Mobile Networks and Applications*, 11(2):161-175, April 2006.
[21] A. C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 1982.